Homework 4

Fhaheem Tadamarry

April 25, 2025

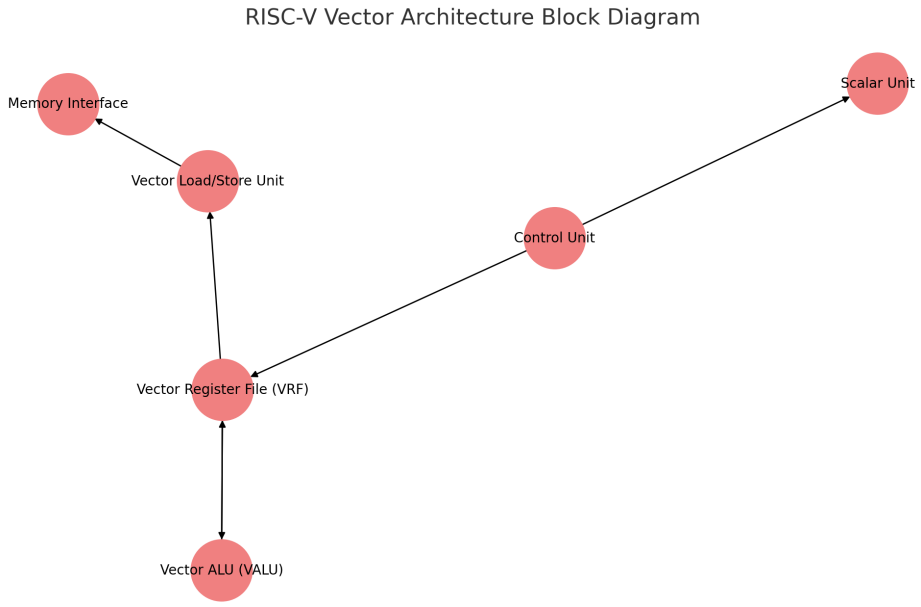
Instruction-Level, Data-Level, and Thread-Level Parallelism

1. **Vector Processing - Basic Idea (15 pts)**
   A **vector processor** operates on entire arrays of data (vectors) with a single instruction, exploiting data-level parallelism (DLP). This contrasts with scalar processors that operate on one data element at a time. Vector processors use pipelined functional units to execute operations across vector elements efficiently.

   Key Features:

   - SIMD architecture (Single Instruction, Multiple Data).
   - Supports vector registers and vector instructions.
   - High throughput due to overlapping execution (pipelining).
   - Reduces instruction-fetch overhead and loop control complexity.

2. **RISC-V Vector Architecture Block Diagram (20 pts)**



   Explanation of Blocks:

- **Vector Register File (VRF):** Holds vector operands and results.
- **Vector Arithmetic Logic Unit (VALU):** Executes vector arithmetic and logic operations.
- **Vector Load/Store Unit:** Manages data transfer between memory and VRF.
- **Scalar Unit:** Handles scalar operations.
- **Control Unit:** Decodes vector instructions, controls the pipeline, and manages vector length.
- **Memory Interface:** Connects to main memory for vector element access.

3. **Vector Processing Techniques (20 pts)**

   a. **Vector Chaining:**
   Allows vector functional units to start executing a new operation before the previous one finishes. The output of one operation can directly feed into the next, reducing pipeline latency.
   Example: Multiply a vector by a scalar and then add another vector without waiting for the full multiplication to complete.

   b. **Strip Mining:**
   Breaks down vector operations into smaller chunks when the vector length exceeds the hardware vector length. Useful for handling loops where data size is larger than the available vector registers.
   Example: If hardware supports vectors of length 64 but data size is 200, the operation is broken into four chunks (64, 64, 64, and 8).

   c. **Vector Stride:**
   Specifies the step between elements in memory for vector operations. Used for accessing non-contiguous memory locations.
   Example: Access every 3rd element in an array: stride = 3.

   d. **Gather and Scatter:**
   - **Gather:** Loads elements from non-contiguous memory addresses into a vector register based on an index array.
   - **Scatter:** Stores elements from a vector register to non-contiguous memory addresses using an index array.
   Useful for irregular data structures like sparse matrices.

4. **Snooping Protocol - Problem 5.1 (20 pts)**

   a. **What are the four states of the MSI protocol?**
   The four states of the MSI protocol are:
   - **M (Modified):** The cache block is modified and is the only valid copy.
   - **S (Shared):** The cache block is clean and may be shared among other caches.
   - **I (Invalid):** The cache block is invalid.
   - **Exclusive (sometimes included in MESI variant, not pure MSI):** Only in MESI, not MSI.

b. **What happens on a processor read hit and miss?**
Read Hit: Data is read from the cache without generating any coherence traffic.
Read Miss: The cache issues a BusRd (bus read) transaction. Other caches may respond with their shared data. If no cache responds, data is fetched from memory. The cache enters the **Shared** state after the miss.

c. **What happens on a processor write hit and miss?**
Write Hit:

- If in **Modified** state: Directly update the cache.
- If in **Shared** state: Issue BusUpgr (bus upgrade) to invalidate other shared copies and transition to **Modified**.

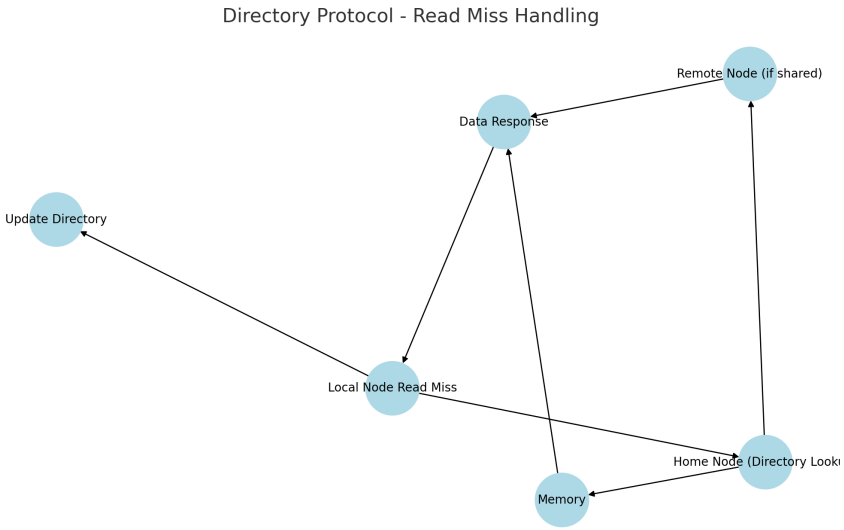
Write Miss: Issues BusRdX (bus read exclusive) to fetch data and invalidate copies in other caches. The cache then transitions to the **Modified** state.

5. **Coherence Protocols Explanation (15 pts)**

a. **Snooping Coherence Protocol:**
In snooping-based coherence, all caches monitor (snoop) the shared bus to determine if they need to take action on memory operations performed by other caches. If one cache writes to a shared block, all other caches snoop this transaction and invalidate their copies if required.
Example: MSI and MESI protocols use snooping mechanisms.

b. **Directory-Based Coherence Protocol:**
Maintains a directory at the home node (associated with each memory block) to keep track of which caches have a copy of the block. Instead of broadcasting to all processors, the home node communicates only with the caches that hold the data, reducing unnecessary traffic.
Example: Scales better than snooping for large multiprocessors since it avoids global broadcasts.

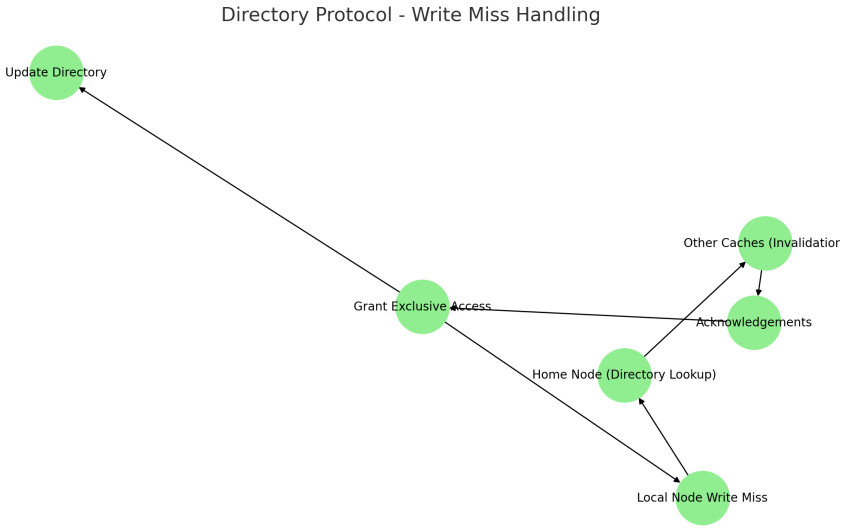
6. **Directory Protocol with State Diagram (10 pts)**
(a) Read Miss Handling:



Explanation:

- A read miss from a local node triggers a request to the home node.
- The home node checks the directory:
  - If the block is not present in any cache, data is fetched from memory.
  - If the block is present in other caches, the data is supplied from one of the remote nodes (if applicable).
- The requesting node is added to the sharing set in the directory.

(b) Write Miss Handling:



Explanation:

- A write miss from a local node triggers a request to the home node.

- The home node sends invalidation messages to all nodes in the sharing set.
- Once invalidations are acknowledged, the home node grants exclusive access to the requesting node.
- The directory updates the owner to the requesting node, clearing the sharing set.